WWW.SCIAM-MAGAZINE.NAROD.RU

Пять несложных программ для различных задач моделирования с помощью случайных чисел

А. К. ДЬЮДНИ

КАКИМ образом можно с помощью пушки измерить площадь пруда? Будем наугад стрелять из пушки по полю, на котором расположен пруд. В результате точки приземления снарядов на поле будут распределены случайно. При достаточно большом количестве выстрелов площадь пруда будет приблизительно равна площади поля, помноженной на число попаданий в пруд и поделенной на общее число выстрелов. Эта, если признаться, несколько глуповатая постановка задачи служит прекрасной иллюстрацией того, какую роль могут играть случайные события при моделировании процессов. Современные ЭВМ — удобное средство моделирования. Правда, элемент случайности в них достигается не стрельбой из пушки, а при помощи датчика случайных чисел.

С философской точки зрения любопытно отметить, что случайные числа, являющиеся чистейшим выражением нашего незнания, легли в основу методов, при помощи которых мы познаем поведение сложных систем. Применение случайных чисел в моделировании проиллюстрируем на примере пяти программ, названных мной несложными пьесами. Воспользуемся случайными числами, чтобы вычислить, например, приближенное значение числа π, частоту появления клиентов в банке, распределение шариков, скатывающихся по доске с колышками, а также выявить временные закономерности, по которым изменяются симпатии избирателей на выборах, и, наконец, показать, как могут возникнуть миллионные очереди.

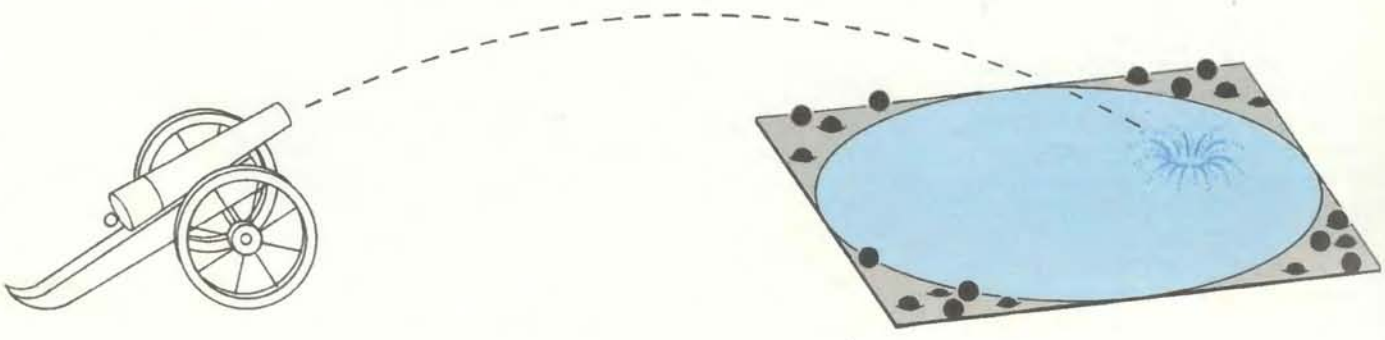
Предлагаемые вниманию читателя программы представляют собой как бы музыкальные пьесы или этюды для начинающих. Обучение игре с компьютером во многом напоминает процесс обучения игре на музыкальном инструменте, и этот процесс может приносить удовлетворение на любой стадии овладения мастерством. Соответственно наша первая пьеса — самая легкая, последнюю же можно, пожалуй, охарактеризовать как несложную. Во всех последующих рассуждениях предполагается, что случайные числа генерируются при выполнении команды *random*. Это будут числа от 0 до 1 с несколькими десятичными разрядами.

Первая пьеса связана с так называемой задачей Бюффона об игле. Граф Луи де Бюффон — французский естествоиспытатель и математик XIX в. Представим себе, что игла многократно и произвольно бросается на дощатый пол. Длина иглы равна половине ширины доски. Задача заключается в том, чтобы найти вероятность пересечения упавшей иглы с линией, разделяющей доски. Ответ оказывается равным $1/\pi$.

Конечно, можно было бы написать программу, моделирующую задачу Бюффона с иглой, однако существует значительно более простой способ вычисления приближенного значения числа π путем моделирования. Воспользуемся для этого идеей, описанной в примере со стрельбой из пушки по полю со случайными попаданиями. Представим себе квадратное поле, заключающее в себе круглый пруд, касающийся всех четырех сторон квадрата. Если стрелять из пушки по этому полю без прицела в определенную точку достаточное количество раз, то отношение числа попаданий в пруд к полному числу выстрелов будет приблизительно равно отношению площади круга к площади квадрата, а именно $\pi/4$.

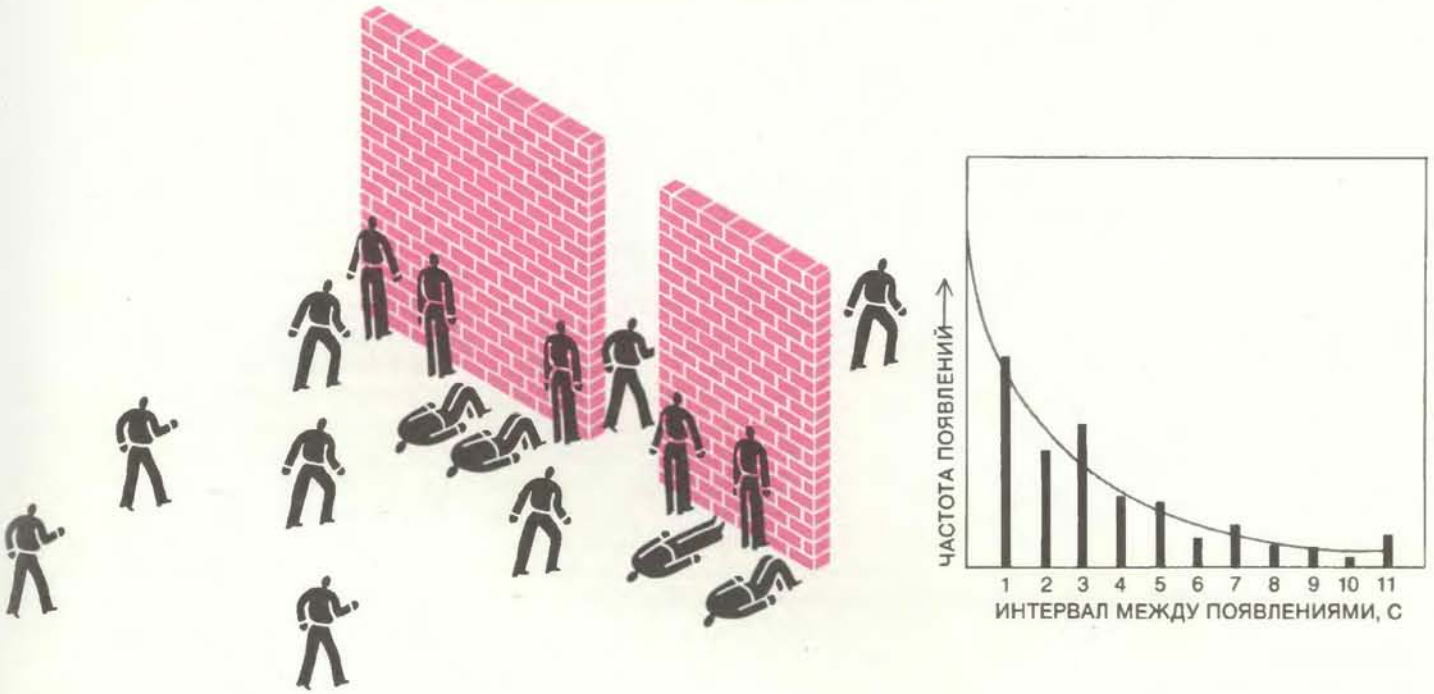
Моделирование можно провести при помощи простой программы, называемой PINT. Представим один квадрант, или четверть поля, скажем северо-восточный, единичным квадратом. Точка (0,0) в юго-западном углу квадранта лежит в центре круга, а точка (1, 1) в северо-восточном углу квадранта представляет собой соответствующий угол поля. Выбрав подряд два случайных числа x и y, можно промоделировать выстрел со случайным попаданием в пределах квадранта (напомним, что каждое случайное число находится в диапазоне от 0 до 1). Попадет ли снаряд в пруд? Чтобы ответить на этот вопрос, вычислим расстояние между точкой (x, y) и центром пруда. Если расстояние меньше или равно 1, то мы регистрируем попадание и увеличиваем значение переменной-счетчика на 1. Программа содержит цикл, при помощи которого «стрельбу» можно продолжать сколь угодно долго. Когда стрельба закончена, помножим отношение числа «попаданий» к общему числу «выстрелов» на 4. Результат должен быть близким к числу π.

СЛЕДУЮЩАЯ пьеса называется «дверь для человечков». Представим себе, что мы наблюдаем за людьми, проходящими через дверь, ведущую в банк. У всех этих людей были веские причины, побудившие их прийти в банк именно в данный момент, и, если бы нам были известны эти причины, мы могли смоделировать моменты появления людей абсолютно точно. Однако мы ничего об этом не знаем. Для подобных задач существует немало моделирующих программ, в которых применяется специальная математическая функция, называемая отрицательным показательным распределением. На объяснение смысла



Площадь круглого пруда можно определить, стреляя по нему из пушки

WWW.SCIAM-MAGAZINE.NAROD.RU



Человечки и закономерность, согласно которой они попадают в проход в стене

этой математической функции и описание довольно изящного способа, позволяющего с ее помощью вычислять моменты появления людей у двери, потребовалась бы не одна страница. Значительно проще объяснить прием «дверь для человечков», использование которого может дать тот же результат.

Дверь для человечков — это проход шириной w в стене, непроницаемой на всем остальном протяжении. Тысячи человечков нескончаемым потоком направляются к стене, и каждую секунду один из них достигает ее в произвольной точке. Те, кому повезет, попадают в проход и идут дальше, остальные натыкаются на стену. Если бы проход в стене был дверью, ведущей в банк, то случайные моменты появления человечков в проходе были бы достаточно близки к наблюдаемым моментам появления реальных клиентов. Чтобы подогнать среднюю частоту появления к реально наблюдаемой, нужно соответствующим образом подобрать ширину прохода в стене.

Рассмотрим программу, моделирующую моменты появления клиентов, под названием ZOMBIE. Стена в программе представлена отрезком прямой линии единичной длины. Цикл типа while генерирует периоды времени между появлениями людей. Каждую секунду, в течение которой ни один человечек не появляется у прохода, переменная-счетчик увеличивается на 1 и происходит обращение к команде *random*. В результате выполнения последней в случайной точке стены появляется человечек. Для удобства и без нарушения общности можно считать, что проход расположен у левого края стены и простирается от точки 0 до точки w. Если случайное число, или точка прибытия очередного человечка, попадает в этот отрезок, то программа выходит из цикла и регистрирует факт прохождения человечка. В противном случае цикл выполняется еще раз. Рано или поздно еще один человечек пройдет через проход, и тогда значение переменной-счетчика укажет, сколько секунд прошло со времени предыдущего появления.

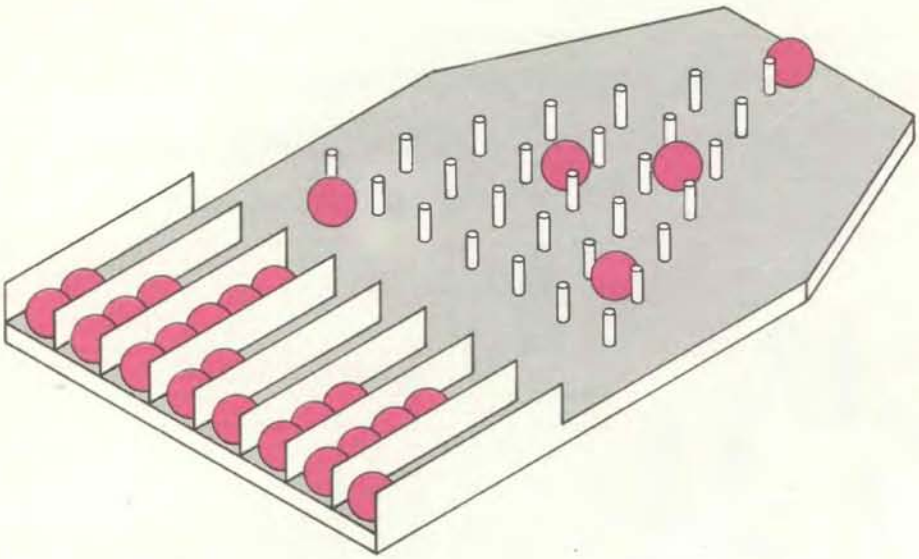
Средний интервал времени между появлениями составляет $1/w$ секунд. Например, если $w = 1/10$, то в среднем один человечек будет проходить через проход каждые $1/(1/10)$, т.е. 10 секунд. Лучше всего программа ZOMBIE работает, когда w достаточно мало. По этой причине подходящим был бы режим (в зависимости от конкретного применения), когда очередной человечек появляется у стены через каждую десятую долю секунды. Чтобы подобрать значение w, дающее хорошие результаты при моделировании, установим средний период времени между появлениями клиентов в банке равным $1/10w$ и отсюда определим w. Выше мы предположили, что проход заключен между точками 0 и w, однако некоторые датчики случайных чисел плохо работают в этом диапазоне. В таком случае можно выбрать другое расположение прохода в стене, например между точками 0,5 и $0,5 + w$ на единичном отрезке.

Любопытно посмотреть, какое распределение интервалов времени между появлениями получается с помощью программы ZOMBIE. Вложив цикл while в другой цикл, подсчитывающий, сколько раз встречается тот или иной интервал между появлениями, можно получить гистограмму, или диаграмму, в виде столбцов, представляющую распределение. Типичная гистограмма изображена на рисунке вверху. На этой же гистограмме для сравнения показано отрицательное показательное распределение. У описанного метода много потенциальных применений для видеоигр, мы же ограничимся рассмотрением лишь одного применения, когда будем обсуждать нашу пятую пьесу.

ТРЕТЬЯ программа моделирует доску Гальтона. Это устройство было названо в честь одного из пионеров статистики, жившего в викторианскую эпоху, сэра Фрэнсиса Гальтона. Оно представляет собой плоскую наклонную поверхность с колышками, расположенными в виде треугольника (см. рисунок на с. 92). Из точки, находящейся над самым верхним колышком, падают шарики, которые затем катятся вниз, натыкаясь по пути на колышки. В самом низу под колышками находятся ячейки, в которые попадают шарики. Когда последний шарик, пройдя между колышками, падает в одну из ячеек, образовавшиеся в ячейках колонки шариков образуют характерную столбчатую диаграмму распределения. Какова форма этой диаграммы, можно выяснить самим, если попытаться решить эту задачу.

В отсутствие настоящей доски Гальтона ее можно смоделировать при помощи программы Galton, состоящей всего из нескольких строчек. Идея, лежащая в основе программы,

WWW.SCIAM-MAGAZINE.NAROD.RU



Доска Гальтона и необычное распределение скатившихся с нее шариков

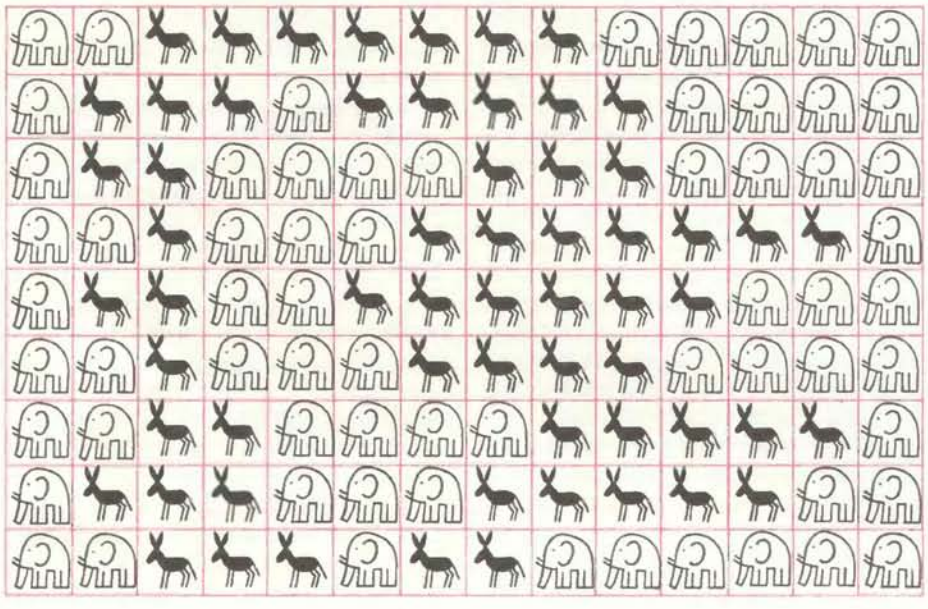
так же как и идея, на которой была основана наша первая пьеса по вычислению числа π, была взята из статьи Дж. Бентли в рубрике «Programming Pearls» («Жемчужины программирования») в журнале «Communications of the ACM» за январь 1984 г.

Спуск каждого шарика моделируется в цикле, где происходит выбор целой последовательности случайных чисел. Каждое из этих случайных чисел определяет, по какую сторону от колышка скатится шарик — справа или слева. Если число меньше или равно 0,5, то моделируемый шарик скатится по левую сторону от колышка, если же число окажется больше 0,5 — по правую сторону. А от какого именно колышка, неважно. Главное — не забыть увеличивать значение счетчика на 1 каждый раз, когда шарик скатывается по правую сторону. Читатель может сам убедиться, что ячейка, в которую попадет шарик, определяется только количеством отскоков в правую сторону. Достоинство программы в том, что в ней устраняется необходимость моделирования поведения шарика при столкновении с каждым колышком. Отсюда следует вывод, что, немного подумав, можно составить простую программу, моделирующую сложный процесс.

Цикл, моделирующий прохождение одного шарика по доске Гальтона, вложен в большой цикл, подсчитывающий число выпускаемых шариков. Каждый раз, когда завершается выполнение внутреннего цикла, ячейка, в которую попадает шарик, должна быть зарегистрирована в массиве c, накапливающем сведения о пополнении ячеек. Элемент массива c (n) увеличивается на 1 всякий раз, когда шарик попадает в ячейку n. После того как был промоделирован спуск, скажем, 1000 шариков, результат, накопленный в массиве c, можно изобразить в виде гистограммы. Если к программе Galton добавить несколько команд вывода на дисплей, гистограмма может быть построена непосредственно компьютером.

Нетрудно сообразить, что при другом расположении колышков на доске Гальтона можно получить распределение другого вида. Предлагаю читателям самим придумать аналогичные устройства независимо от того, существуют ли они реально или только в воображении.

Четвертая несложная пьеса связана с двумя новыми интересными играми, моделирующими динамику изменения симпатий избирателей во время предвыборной кампании. Эти игры были недавно изучены П. Доннели из Университетского колледжа Суонси в Уэльсе и Д. Уэлшем из Оксфордского университета. Клетки прямоугольной решетки в исходном состоянии раскрашены в белый и черный цвета в случайном порядке. Каждый цвет представляет политическую ориентацию избирателя, занимающего данную клетку. Например, один цвет может означать республиканца, другой — демократа. Допустим, что на каждый «тик» и «так» часового механизма случайным образом выбирается избиратель и его политическая ориентация может быть изменена. Для этого произвольно выбирается один из восьми соседей данного избирателя и политические симпатии соседа передаются этому избирателю независимо от его прежних убеждений.



Одна из ситуаций, сложившихся в игре «голосование»

РЕШЕТКА замыкается сама на себя аналогично океану планеты Акватор (см. статью в рубрике «Занимательный компьютер», журнал «В мире науки» № 2 за 1985 г.). Верхний край решетки отождествляется с нижним краем, а левый край — с правым. Поэтому избиратель, находящийся в клетке на краю решетки, имеет трех соседей на противоположной стороне. После каждого случайного изменения политической ориентации одного избирателя устраиваются «выборы», однако это не обязательный элемент модели и его можно исключить, если играющий не хочет следить за динамикой результатов.

По мере того как реализуется эта незамысловатая модель, происходят странные вещи. Сначала на решетке образуются большие блоки одного цвета. Блоки представляют территориальные области, где политические ориентации всех избирателей совпадают. Затем блоки начинают переме-

WWW.SCIAM-MAGAZINE.NARODRU

шаться по решетке, пока не остаются только два блока, борющихся между собой. Наконец, и эта система из двух блоков рушится, и все начинают голосовать одинаково. Меньший блок исчезает бесследно, а за ним и вся система. Игра наводит на философские размышления.

Короткая программа VOTERS моделирует описанный процесс во всех деталях. Она содержит массив тех же размеров, что и решетка. В коротком вычислительном цикле выбираются три случайных целых числа, дающие соответственно случайный номер строки, случайный номер столбца и код случайного соседа на каждом такте часов. Три целых числа получаются путем умножения величины, возвращаемой командой *random*, на постоянную и удаления дробной части при выполнении команды *integer*. Если, например, решетка содержит 50 рядов (строк), то выбор случайного индекса строки i будет алгоритмически записан как

$$i \leftarrow \text{integer}\ (50 * \text{random}).$$

Случайный индекс столбца j вычисляется аналогичным образом. Выбор программой соседа клетки (i, j) зависит от того, какое из целых чисел от 0 до 7 было выбрано. После того как были случайно выбраны клетка и ее сосед, программа VOTERS заменяет значение элемента массива (i, j) значением, соответствующим соседней клетке.

Описанные шаги алгоритма заключены в цикл. Количество повторений цикла удобно сделать переменной величиной. Тогда можно экспериментировать с количеством изменений мнения избирателей. Пределы изменения индексов могут варьировать и в других программах, рассматриваемых в настоящей статье. Для этого количество повторений цикла устанавливается равным n, а желаемое значение n вводится с клавиатуры перед выполнением программы.

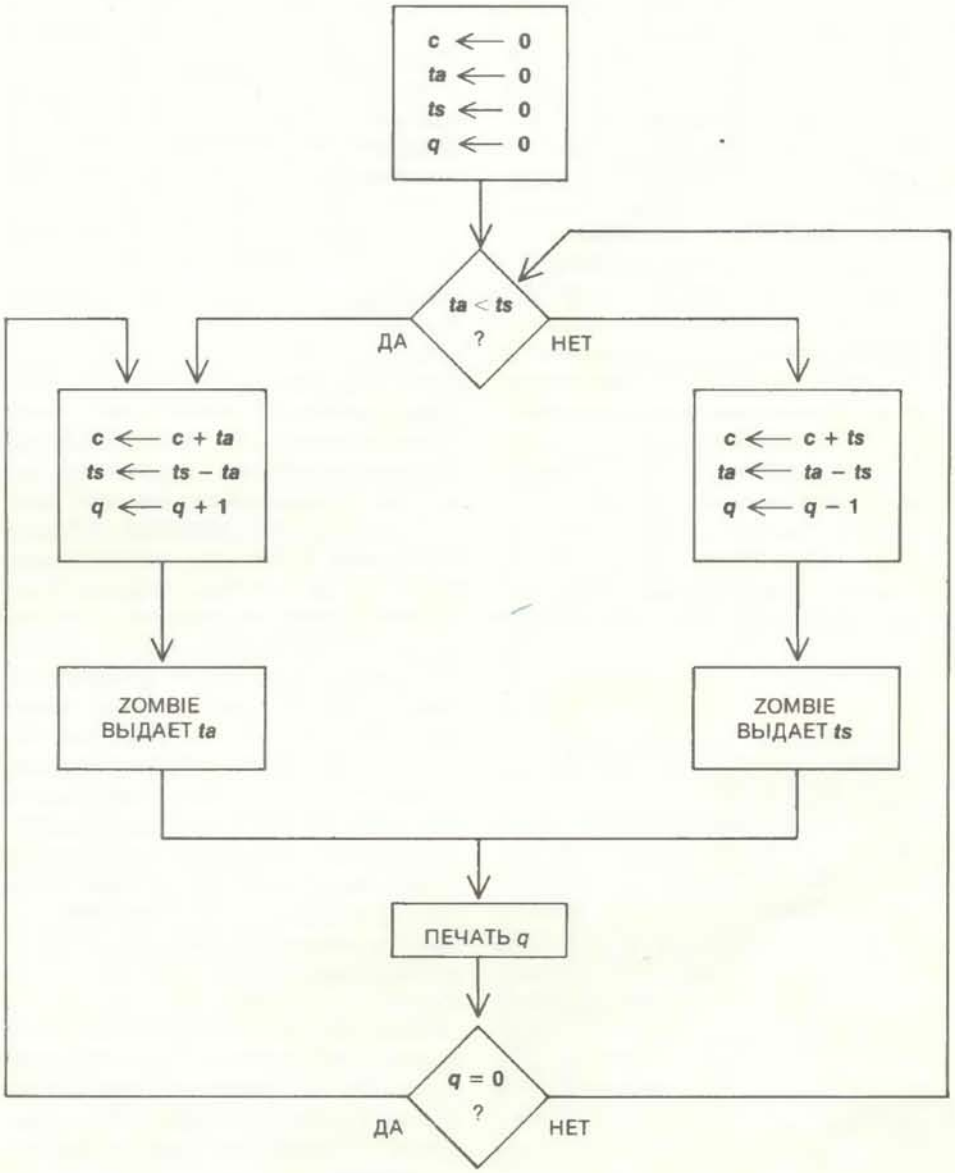
Графическое представление решетки нетрудно реализовать в основном вычислительном цикле программы. Даже при использовании только двух символов (скажем, точки и звездочки) получается впечатляющее зрелище. Читатели, которых не остановит солидарность мнений избирателей в исходе игры «голосование», возможно, захотят реализовать и ее обратный вариант, который Уэлш и Доннели называют игрой «антиголосование». В отличие от «прямого» варианта в игре «антиголосование» случайно выбранный избиратель принимает мнение, противоположное мнению его случайно выбранного соседа. Сохранится ли солидарность избирателей при этих условиях?

ПЯТАЯ, и последняя, пьеса дает пищу для размышления человеку, стоящему в очереди. Если люди будут уходить из начала очереди в среднем с той же скоростью, с какой другие прибывают в конец, то будет ли размер очереди оставаться приблизительно одним и тем же? По-видимому, нет. Согласно теории очередей, такая очередь не имеет конечной ожидаемой длины. Подобные очереди я наблюдал в банках.

Программа QWING моделирует динамику простой очереди. Люди прибывают в конец очереди в соответствии с той же закономерностью, какая порождается уже знакомой нам программой ZOMBIE. В начале очереди находится человек, обслуживаемый в данный момент. В программах, моделирующих очередь, иногда предполагается, что длительности обслуживания распределены так же, как интервалы времени между двумя случайными прибытиями клиентов. Когда в банке я вношу вклад на свой счет, процедура обслуживания обычно не занимает много времени. Значительно дольше длится разговор со служащим банка, если клиент выясняет, куда делись деньги с его счета. Поэтому для получения случайных длительностей обслуживания программа QWING обращается к ZOMBIE как к подпрограмме.

Ход времени обрабатывается в программе QWING на основе метода критического события. Показания моделируемых часов постоянно изменяются к моменту следующего события, будь то уход из очереди или прибытие нового человека. Показания часов обозначим через c. Потребуются еще три переменные: ts — промежуток времени, оставшийся до момента завершения текущей процедуры обслуживания, ta — промежуток времени до следующего прибытия и q — длина очереди (см. рисунок внизу).

Вначале значения ts и ta устанавливаются равными 0, а $q = 1$. Затем программа входит в свой основной цикл, причем количество его повторений задается с клавиатуры специальной



Блок-схема программы **QWING**

WWW.SCIAM-MAGAZINE.NAROD.RU

командой. Цикл содержит команду условного перехода. Она проверяет, меньше ли величина *ta* величины *ts*. Если это так, то показания часов увеличиваются на величину *ta*, потому что критическим событием в данном случае является следующее прибытие. Соответственно величина *ts* уменьшается на то же число, и, поскольку критическим событием является прибытие, *q* увеличивается на 1. На заключительном шаге алгоритма в работу включается программа ZOMBIE, чтобы выдать случайный промежуток времени до следующего прибытия.

В начале цикла может выясниться, однако, что *ta* не меньше *ts*. В этом случае аналогичные операции выполняются в другой ветви программы, с той лишь разницей, что *ta* и *ts* меняются ролями. Критическим событием является следующее обслуживание. Поэтому *q* уменьшается на 1. Здесь программа ZOMBIE выдает следующее значение величины *ts*. В конце цикла обе ветви сходятся воедино, и производится проверка значения *q*. Если $q = 0$, то в ожидании следующего прибытия программа переходит к первому сегменту. В противном случае она возвращается в начало цикла, где сравниваются значения *ta* и *ts*.

Программа *QWING*, очевидно, длиннее других, но она еще достаточно коротка, чтобы охарактеризовать ее как «простую». Будет, наверное, лучше, чтобы пользователь мог сам указать такие величины, как средняя продолжительность обслуживания и средний промежуток времени между двумя прибытиями клиентов. Благодаря этому каждый может самостоятельно проверить основные результаты теории очередей. Если средний промежуток времени между двумя прибываниями больше средней продолжительности обслуживания, то ожидаемая длина очереди будет конечной величиной. Чем больше разность между этими величинами, тем более «конечной» становится длина очереди. Когда средние величины находятся в противоположном отношении, ожидаемая длина очереди является бесконечной — понаблюдайте, как она растет. Когда средние величины в точности равны друг другу, ожидаемая длина очереди не является ни конечной, ни бесконечной. Как такое может быть? Проведите эксперимент (или хорошенько подумайте) и вы получите ответ на этот вопрос.

СОСТОЯНИЕ	ВХОД	СЛЕД. СОСТОЯНИЕ	ВЫХОД	НАПРАВЛЕНИЕ
A	0	B	1	ПРАВОЕ
	1	C	1	ЛЕВОЕ
B	0	A	0	ЛЕВОЕ
	1	D	0	ЛЕВОЕ
C	0	A	1	ЛЕВОЕ
	1	H	1	ЛЕВОЕ
D	0	B	1	ЛЕВОЕ
	1	E	1	ПРАВОЕ
E	0	D	0	ПРАВОЕ
	1	B	0	ПРАВОЕ

Бобер-работяга с пятью состояниями?

ЧИТАТЕЛИ, склонные к анализу любопытных ситуаций, могут усложнить программу QWING, чтобы смоделировать одно из новшеств, которое было введено в банках более десяти лет назад. Вместо того чтобы ожидать в различных очередях, к каждому окошку в отдельности, клиенты становились в одну общую очередь. Эта система обладает несколькими преимуществами, в чем можно убедиться с помощью моделирующей программы, которая учитывает основные свойства коллективной очереди. Еще более сложная программа может смоделировать сравнительно более недавнее нововведение: теперь каждый обслуживающий клиентов работник банка находится в очереди к тому, кто обслуживает их самих. Эта новая система оказывается очень выгодной для крупных банков.

По поводу всех этих рассуждений о случайных числах возникает вопрос: как может компьютер, представляющий собой детерминированную машину, генерировать случайные числа, по природе своей явно не детерминированные? Ответ прост — не может. Однако компьютер может генерировать числа, которые *похожи* на случайные. В самом деле, возьмем произвольное число, умножим его на m, прибавим k и возьмем остаток от деления полученной величины на p. Остаток вновь подвергнем той же процедуре. Раз за разом этот алгоритм будет выдавать так называемые псевдослучайные числа. Однако рано или поздно последовательность результатов, выдаваемых алгоритмом, должна повториться, потому что существует только p возможных остатков. Более того, если величины m, k и p не подобраны самым тщательным образом, то получаемые псевдослучайные числа не удовлетворяют даже самым простым тестам на случайность.

Зная об этих недостатках, программисты-теоретики пытались придумать лучший метод. Много лет назад Д. Кнут из Станфордского университета изобрел алгоритм настолько «случайно превосходный» и «превосходно случайный», что генерируемые им числа, казалось, были достойны эпитета «сверхслучайные». Алгоритм Кнута состоит из 12 шагов. Для заданного исходного числа X на первых двух шагах выбираются два разряда этого числа. Выбранные разряды определяют, сколько раз алгоритм должен быть повторен и к какому из следующих 10 шагов алгоритма следует перейти. Каждый из этих 10 шагов представляет собой свой, отличный от других метод вычисления нового случайного числа по заданному старому. Перед тем как Кнут испытал свой алгоритм в действии, ему казалось, что «он обеспечит производство по крайней мере бесконечного количества невероятно случайных чисел». К его удивлению, «когда алгоритм был впервые испытан на машине, он почти немедленно сошелся на десятиразрядном числе 6065038420, которое, в силу исключительно редкого совпадения, порождало само себя». Отсюда Кнут вывел простую, хотя, может быть, интуитивно неочевидную модель: «Случайные числа не должны генерироваться случайно выбранным методом. Эта процедура должна быть теоретически обоснована».

Если численные методы получения случайных чисел окажутся неудовлетворительными, можно всегда воспользоваться предложением, когда-то сделанным Аланом М. Тьюрингом. Он указал на то, что датчик случайных чисел может быть основан на радиоактивном источнике. Читатели могут с интересом ознакомиться с развитием этой идеи, прочтя статью F. M. Mims III, "The Computer Scientist: Random Numbers" в ноябрьском номере журнала "Computers and Electronics" за 1984 г.

СО ВРЕМЕНИ появления статьи о программе Рэктер, любительнице салата, возникло больше историй, чем она создала сама. Теперь я уже провел достаточно много бесед с программой Рэктер и понял, что ее умопомешательство значительно серьезнее, чем я вначале предполагал.

Дж. Ауенз из Стэйтен-Айленда, шт. Нью-Йорк, занимается распространением этой программы. Он сообщил, что один экземпляр был заказан для обучения практикантов-психиатров, часто беседующих с пациентами-шизофрениками. Другой экземпляр достался владельцу, которого Рэктер чуть было не свела с ума. Эта «сумасшедшая», по его словам, программа никак не хотела заканчивать беседу, настаивая на дальнейшем обсуждении высказанного ею предположения, что святой Петр был на самом деле атеистом. Я сочувствую этому человеку, потому что сам испытал на себе всю гамму эмоций, бе-

WWW.SCIAM-MAGAZINE.NAROD.RU

седуя с Рэктер. Иногда я смеялся, иногда скучал, а иногда даже приходил в бешенство. Но стоит ли так реагировать, ведь в конце концов Рэктер — всего лишь программа!

Однако для семьи Ауензов Рэктер — это нечто большее, чем программа. Один клиент в своем письме шутливо спросил, может ли Рэктер «перевоплотиться» в восьмидюймовый дискет. Решили спросить у самой программы, и она ответила: «Если мысль о том, что «мы сможем перевоплотить меня в восьмидюймовый дискет» придет в голову пессимисту, то он может подумать, что это действительно пессимизм». Позже выяснилось, что в программном хозяйстве Ауензов отсутствовали средства подобного перевоплощения.

Другой раз Терри Ауенз спросил Рэктер, сколько денег ему следует вложить в рекламу. «Очень много денег, Терри, потому что люди поверят ей», — ответила Рэктер.

В прошлом номере журнала я упомянул о новом кандидате на звание «бобер-работяга с пятью состояниями» (см. также статью «Охота с компьютером на бобра-работягу, или поиск самой трудолюбивой машины Тьюринга», «В мире науки», 1984, № 10). В декабре прошлого года Дж. Уинг из Бронкса, шт. Нью-Йорк, смоделировал машину Тьюринга, которая выдала 1915 единиц прежде, чем остановилась. Машина Уинга представлена в таблице, помещенной на предыдущей странице.

Если нам нужно определить, что́ машина сделает, например, в состоянии B, рассмотрим строку, помеченную этим символом. Строка подразделяется на верхнюю и нижнюю части, указывающие ответы машины на 0 и 1 соответственно. Если машина встречает на ленте 1, то она переходит в состояние D, записывает на ленте 0 и передвигается на одну клетку влево. Буква H в таблице обозначает останов машины.

Уинг, программирующий для манхаттанской оптической компании, прочтя статью о машине Тьюринга, решил заняться поисками «бобра-работяги с пятью состояниями». Он воспользовался микропроцессором Z-80, для которого написал программу на языке ассемблера. Эта программа управляла в свою очередь другой машиной, моделирующей машину Тьюринга. Модель обошлась Уингу меньше чем в 100 долл. Созданный им автомат выполняет 7 млн. переходов машины Тьюринга в секунду. Каждый переход сводится просто к обращению к таблице, подобной той, что приведена на рисунке на с. 94. Уинг полон решимости найти автомат типа «бобер-работяга с пятью состояниями». Может быть, рассматриваемая машина уже удовлетворяет всем необходимым требованиям? Уинг получил свои первые результаты после того, как машина работала в течение месяца. Насколько мне известно, она все еще работает.

А. Брейди из Университета шт. Невада в г. Рено охарактеризовал машину Уинга как «поразительную». Каковы шансы, что мы откроем автомат с шестью состояниями? «Это совершенно исключено», — говорит Брейди.

Наука и общество

Как убить рак

В КОНЦЕ XIX в. нью-йоркский врач Уильям Б. Коли обратил внимание, что у онкологических больных бактериальная инфекция иногда приводит к тому, что опухоль регрессирует: уменьшается в размерах или даже совсем исчезает. Коли стал лечить рак препаратами убитых бактерий, которые действовали подобно естественной инфекции. Нередко это давало хорошие результаты, причем в отличие от современных химиотерапевтических и лучевых методов не наносило вреда здоровым, непораженным раком тканям.

Долгие годы механизм обратного развития опухолей, описанный Коли, оставался загадкой. В настоящее время мы приблизились к пониманию одной из причин этого явления. Идентифицировано вещество, играющее ключевую роль в процессе регрессии; оно изучено и недавно получено с помощью методов генетической инженерии.

В 1975 г. Л. Олд и его коллеги (Центр по изучению рака им. Слоуна — Кеттеринга) сообщили о том, что такое вещество существует. Они воспроизвели в эксперименте условия, в которых Коли наблюдал регрессию опухолей. Исследователи инъецировали мышам с опухолями эндотоксин (препарат, выделяемый из бактерий некоторых штаммов), в результате чего опухоли отмирали — наблюдался так называемый геморрагический некроз. Сотрудники Олда обнаружили, что эффект эндотоксина является опосредованным: под влиянием эндотоксина в организме мыши синтезируется вещество, которое и воздействует на опухоль. Это вещество было названо фактором некроза опухолей (ФНО). ФНО действует избирательно. При воздействии мышиной сыворотки, содержавшей ФНО, на культуру злокачественных клеток рост ее останавливался и клетки погибали; на нормальные клетки такая сыворотка не влияла.

Олд и его коллеги выявили и клетки, продуцирующие ФНО. Перед введением мышам эндотоксина некоторых из них заражали БЦЖ. БЦЖ (традиционное от BCG — сокращения слов bacillus Calmette — Guérin, что значит «бацилла Кальметта — Жерена») — это аттенуированный штамм микобактерий туберкулеза, т.е. живые, но ослабленные палочки Коха; известно, что они дают противоопухолевый эффект. Измеримое количество ФНО продуцировалось лишь у тех животных, которым вводили БЦЖ (хотя и в организме незараженных мышей количество ФНО было достаточным для того, чтобы начался геморрагический некроз). Инъекция БЦЖ вызывала у них быструю пролиферацию макрофагов — клеток иммунной системы, выполняющих функцию «мусорщиков». Исследователи сделали вывод, что у мышей ФНО синтезируется, скорее всего, в макрофагах. Впоследствии выяснилось, что и у человека ФНО образуется в макрофагах и других клетках иммунной системы.

Однако оставалось непонятным, в чем состоит воздействие ФНО на опухолевые клетки и каким образом он отличает злокачественные клетки от нормальных. Наверняка фактор некроза действует совместно и согласованно с другими веществами, образуемыми иммунной системой. В 1983 г. Олд и другие показали, что эффект ФНО проявляется наиболее ярко в присутствии интерферона. Глубже разобраться в этом исследователям не удалось — главным образом потому, что в их распоряжении не было достаточного количества очищенного ФНО.

Теперь проблема очистки решена. В журнале "Nature" опубликовано сообщение группы Д. Гёдела (Genentech, Inc.) о том, что клонирован ген, кодирующий человеческий ФНО; таких же результатов добились в лабораториях двух других компаний, занимающихся биотехнологией. Открывается возможность подробно изучить молекулярную структуру ФНО и механизм, посредством которого он «узнает» злокачественные клетки. Можно будет также исследовать, какую роль играет фактор некроза в нормальных клетках. Клонирование гена ФНО позволит испытать его в клинике и проверить наблюдения Коли.